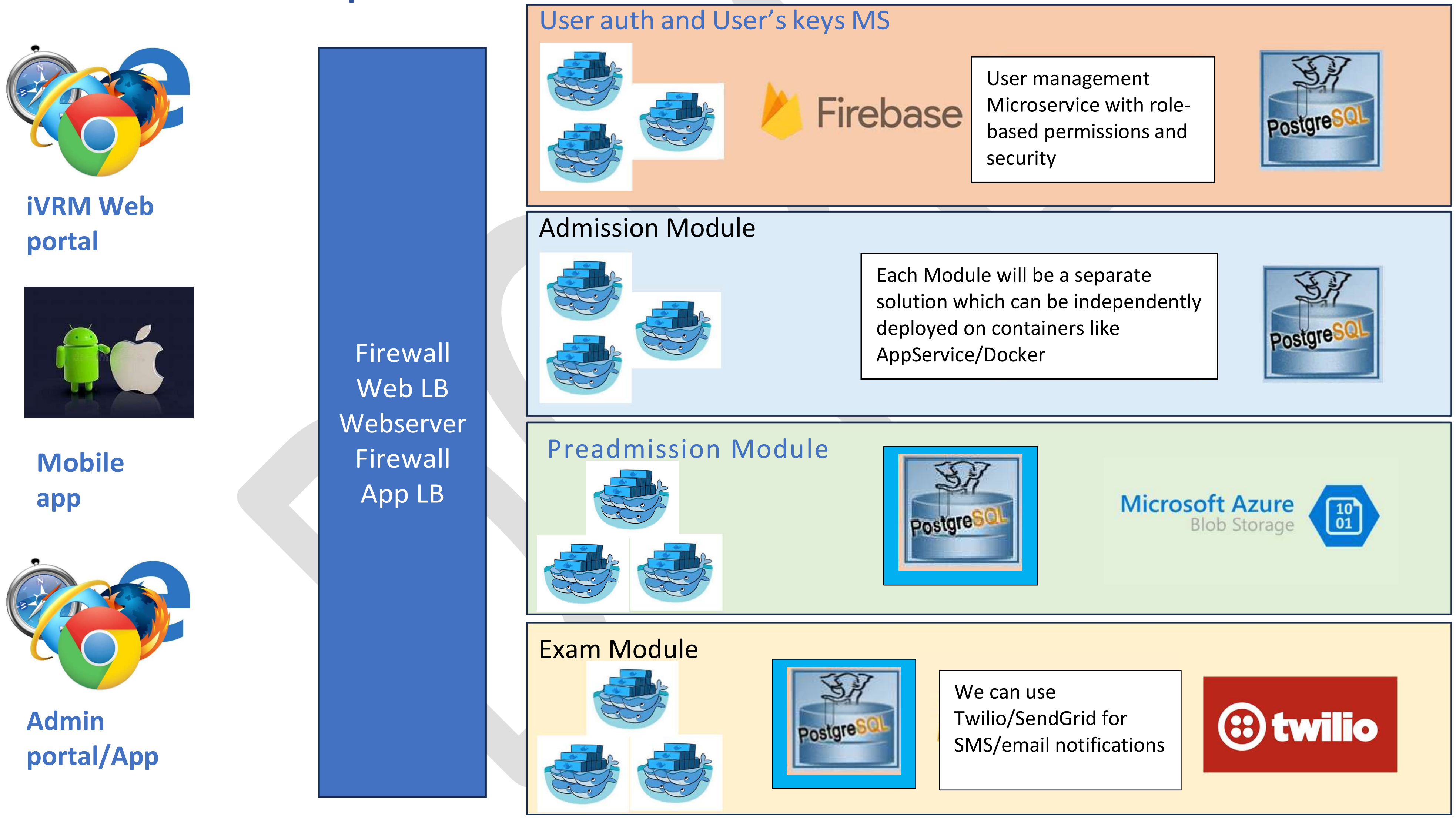
iVRM- Proposed Architecture Sarvaantar AI Solutions Pvt Ltd

Sarvaantar AI Solutions Pvt Ltd

Basic architecture representation



User management Microservice with role-based permissions and security

Each Module will be a separate solution which can be independently deployed on containers like AppService/Docker

We can use Twilio/SendGrid for SMS/email notifications

Different Microservices

➢ User auth and User's keys MS

❑ User and Authentication

- User management
- Authentication
- 2-FA integration with Firebase/Inbuilt

❑ Authorization

- Session management
- Role/Persona management
- Role/Persona enforcement

❑ Key management

- Generate unique private and public keys
- Storing and retrieving keys from azure vault

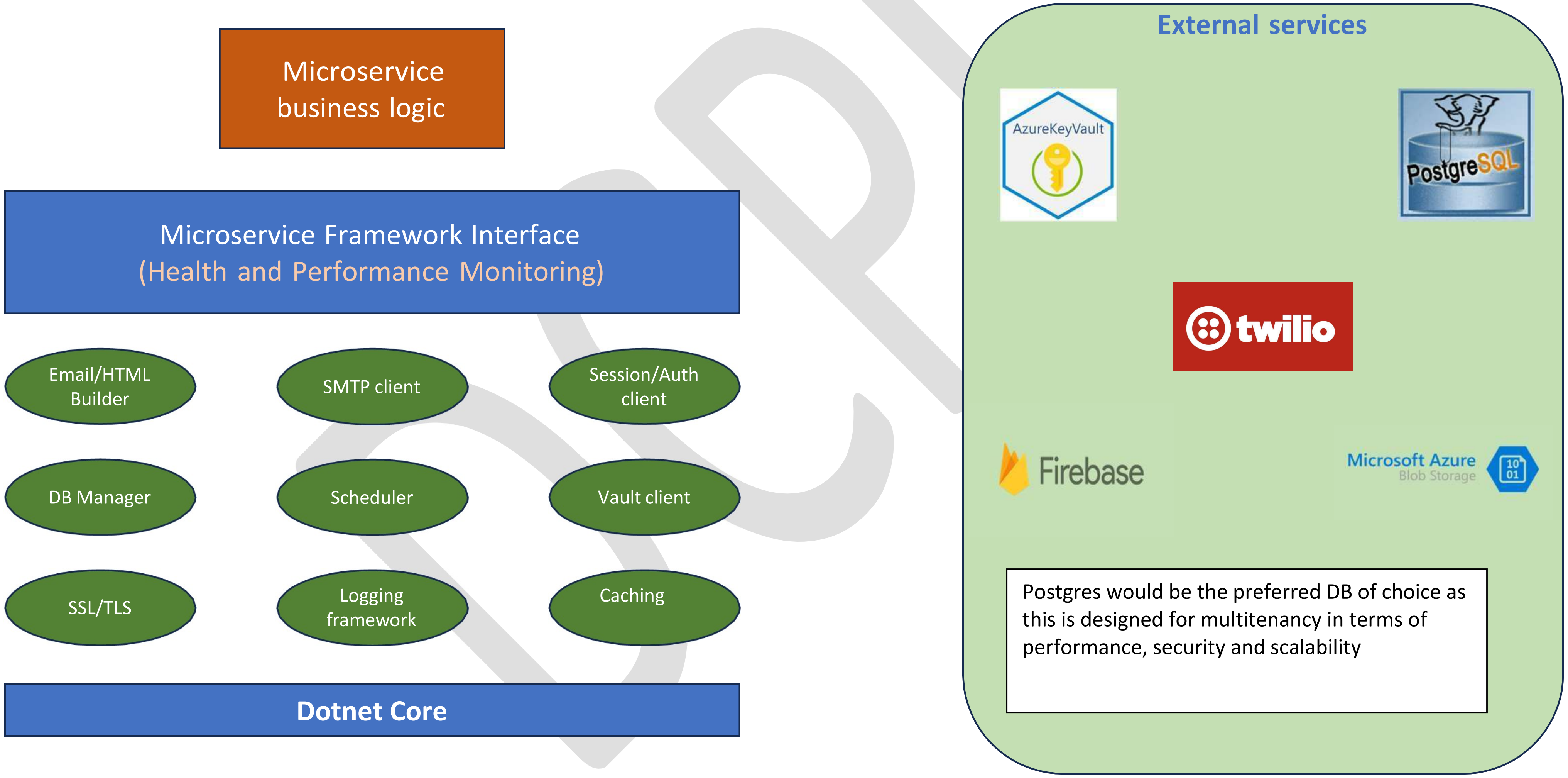
Different Module Level Microservices

➢ Admission Module

➢ HRMS Module

➢ Inventory Module

Service Architecture Representation



Postgres would be the preferred DB of choice as this is designed for multitenancy in terms of performance, security and scalability

Recommended TechStack

UX – FIGMA
UI – NextJS
API/Microservice –Dotnet core
Database – Postgres
Cloud – Azure
Mobile App – Flutter (Hybrid)

Technology Stack - UI

- **Framework**: React.js with Next.js for server-side rendering
- **State Management**: Redux for application state
- **Styling**: Tailwind CSS for responsive design
- **UI Components**: Shad or Material UI or Chakra UI for consistent interface elements
- **Authentication**: JWT tokens

Key Features for User Experience

- **Responsive Design:** Ensure the home page works seamlessly on all devices.
- **Fast Loading:** Optimize images, videos, and code for faster load times.
- **Accessibility:** Follow WCAG guidelines for accessibility (e.g., alt text, keyboard navigation).
- **Personalization:** Show personalized recommendations based on user behaviour

System Overview

Core Capabilities

- Multi-tenant architecture supporting white-label partner portals
- Unified API integration for content providers
- Comprehensive user management for students and administrators
- End-to-end payment processing with centralized gateway
- Advanced reporting and analytics
- Accounting system integration

User Experience (UX) Design

- **Navigation:** Easy navigation with a clear menu structure, breadcrumbs, and a search bar.
- **Responsive Design:** Ensure the portal is mobile-friendly and works seamlessly across devices (desktop, tablet, mobile).
- **Accessibility:** Follow WCAG guidelines to make the portal accessible to users with disabilities.
- **Performance:** Optimize images, use lazy loading, and minimize HTTP requests to ensure fast loading times.

Design Philosophy

The frontend will follow a modern, responsive design using a component-based architecture that enables:

- **Consistent branding with white-label customization**
- **Accessibility** across all devices and platforms
- **Intuitive navigation** with minimal learning curve
- **Performance optimization** for fast page loads even with rich content

API Integration Framework

Architecture Pattern

- RESTful API design with OpenAPI specifications
- API Gateway for centralized management
- JWT for secure authentication
- Rate limiting and throttling for stability

Security Implementation

Data Protection

- In-transit encryption with TLS/SSL
- At-rest encryption for all databases and storage
- Field-level encryption for sensitive information

Access Control

- Role-based access control (RBAC)
- Multi-factor authentication for administrative access
- IP restriction for administrative functions
- Principle of least privilege implementation

Compliance Measures

- GDPR compliance for user data
- PCI DSS compliance for payment processing

Integration Touchpoints

Analytics Integration

- Google Analytics for user behaviour
- Custom event tracking for business metrics
- Data warehouse integration for advanced analytics

Search Engine Optimization

Technical SEO Implementation

- Server-side rendering for core pages
- Structured data markup (Schema.org)
- XML sitemap generation
- Canonical URL implementation
- Mobile optimization

Content SEO Strategy

- Content-specific landing pages
- Educational content hub
- Targeted keyword optimization
- Meta tag management system

Scalability & Performance

Horizontal Scaling

- Auto-scaling configuration for application tiers
- Read replicas for database scaling
- Distributed caching strategy

Performance Optimization

- CDN implementation for static content

- Image and video optimization pipeline
- Lazy loading for content-heavy pages
- API response caching

Backend Architecture

The backend architecture should be scalable, secure, and capable of handling multiple API integrations. Here's a detailed architecture:

Azure Infrastructure

- Appservice/Docker: Host the web application on Appservice instances for scalability and flexibility.
- Azure Postgres as Service: Use APAS for database management (PostgreSQL) to store student, course, and transaction data.
- Azure Blob: Store static assets (images, videos, documents) and content materials.
- Azure Functions: Use serverless functions for handling API requests, payment processing, and other backend tasks.
- Azure API Gateway: Manage and secure API integrations with whitelabel partner portals.
- Azure CDN: Use a CDN to deliver content quickly to users globally.
- Azure IAM: Implement Identity and Access Management to secure access to Azure resources.
- Application Insights: Monitor and log application performance and errors**.**

Future Enhancements

- AI-Powered Search: Integrate AI for smarter recommendations.
- Gamification: Add badges or progress bars for user engagement.
- Live Chat: Integrate a chatbot for instant support.
- Multi-Language Support: Add support for multiple languages.

Microservice ready framework

- Built on Dotnet Core 8.0
- C# 10
- PostgreSQL 15.3 and above
- Visual Studio IDE 2022